IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of:	§	Group Art Unit: 2436
	§	
Hans Eberle	§	Examiner: Johnson, Carlton
Nils Gura	§	
Lawrence Spracklen	§	Atty. Dkt. No.: 6000-31500
Sheueling Chang-Shantz	§	
Leonard Rarick	§	
	§	
Serial No.: 10/789,311	§	
	§	
Filed: February 27, 2004	§	
	§	
For: Method and Apparatus for	§	
Implementing Processor	§	
Instructions for Accelerating	§	
Public-Key Cryptography	§	

APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir/Madam:

Further to the Notice of Appeal filed October 22, 2010, Appellants present this Appeal Brief. Appellants respectfully request that the Board of Patent Appeals and Interferences consider this appeal.

I. REAL PARTY IN INTEREST

The subject application is owned by Sun Microsystems, Inc., which is now Oracle America, Inc.

II. RELATED APPEALS AND INTERFERENCES

No other appeals, interferences or judicial proceedings are known which would be related to, directly affect or be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF CLAIMS

Claims 1-67 are pending in the application and stand finally rejected. The rejection of claims 1-67 is being appealed. A copy of claims 1-67 is included in the Claims Appendix herein below.

IV. STATUS OF AMENDMENTS

No amendments have been submitted subsequent to the final rejection.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Independent claim 1 is directed to a method implemented in a device supporting a public-key cryptography application. (*See, e.g.*, p. 1, paragraph [1002].) The method includes a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures. (*See, e.g.*, FIG. 4A, which illustrates circuitry for implementing a *umulxc* instruction; p. 2, paragraph [1006]; p. 3 paragraph [1008]; and pp. 11-12, paragraph [1068].) The method also includes the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application. (*See, e.g.*, FIG. 4A; pp. 11-12, paragraph [1068] and Table 1.) The currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits. (*See, e.g.*, p. 12, Table 1; and pp. 12-13, paragraph 1070].) The first partial result represents the high order bits summed with low order bits of a result of a first number multiplied by a second number. (*See, e.g.*, p. 2, paragraph [1006].) The summing of the high order bits is performed during multiplication of the first number and the second number. (*See, e.g.*, p. 2, paragraph [1006].) The summing and at least a portion of the multiplication are performed in the second arithmetic circuit. (*See, e.g.*, p. 2, paragraph [1006]; and p. 3 paragraph [1008].) The method also includes storing the first partial result and using the stored first partial result in a subsequent computation in the public-key cryptography application. (*See, e.g.*, pp. 11-12, paragraphs [1068-1069] and Table 1.)

Independent claim 21 is directed to a method implemented in a device supporting a public-key cryptography application. (*See, e.g.*, p. 1, paragraph [1002].) The method includes a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of

arithmetic structures. The method also includes supplying a third number to the second arithmetic circuit. (*See, e.g.*, FIG. 4B, which illustrates circuitry for implementing a *umulxck* instruction; pp. 2-3, paragraph [1007]; p. 3 paragraph [1009]; and pp. 13-14, paragraph [1072] and Table 2.) The method also includes the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application. (*See, e.g.*, FIG. 4B; and pp. 13-14, paragraph [1072] and Table 2.) The currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits (*See, e.g.*, pp. 13-14, paragraph [1072] and Table 2; and pp. 14-15, paragraph [1076].) The first partial result represents the high order bits summed with low order bits of a result of a first number multiplied by a second number and with the third number. (*See, e.g.*, pp. 2-3, paragraph [1007].) The summing is performed during multiplication of the first number and the second number. (*See, e.g.*, pp. 2-3, paragraph [1007].) The summing and at least a portion of the multiplication are performed in the second arithmetic circuit. (*See, e.g.*, pp. 2-3, paragraph [1007]; and p. 3 paragraph [1009].) The method also includes storing the first partial result and using the first partial result in a subsequent computation in the public-key cryptography application. (*See, e.g.*, pp. 13-14, paragraph [1072] and Table 2.)

Independent claim 38 is directed to a processor configured to support public-key cryptography applications. (*See, e.g.*, the Title; and pp. 7-8, paragraph [1056].) The processor includes a first plurality of arithmetic structures configured to generate high order bits for an arithmetic instruction of the processor's instruction set in a public-key cryptography application that includes a multiplication operation. The processor also includes a second plurality of arithmetic structures configured to generate low order bits of the arithmetic instruction. The second arithmetic structures are further configured to receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic instruction in the public-key cryptography application and to generate a first partial result of a currently executing instance of the arithmetic instruction. (*See, e.g.*, FIG. 4A, which illustrates circuitry for implementing a *umulxc* instruction; p. 2, paragraph [1006]; p. 3 paragraph [1008]; and

pp. 11-12, paragraph [1068] and Table 1.) The arithmetic instruction does not include an explicit source operand for specifying the high order bits. (*See, e.g.*, p. 12, Table 1; and pp. 12-13, paragraph 1070].) The first partial result represents the high order bits summed with low order bits of a multiplication result of the multiplication operation. (*See, e.g.*, p. 2, paragraph [1006].) The processor further includes a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application. (*See, e.g.*, FIG. 4A, extended carry register (exc) 403; pp. 11-12, paragraphs [1068-1069] and Table 1.)

Independent claim 53 is directed to a processor configured to support public-key cryptography applications. (*See, e.g.*, the Title; and pp. 7-8, paragraph [1056].) The processor includes a first plurality of arithmetic structures configured to generate high order bits for an arithmetic instruction of the processor's instruction set in a public-key cryptography application that includes a multiplication operation of a first and a second number. The processor also includes a second plurality of arithmetic structures configured to generate low order bits of the arithmetic instruction. The second arithmetic structures are configured to receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic instruction, and to receive a third number. (*See, e.g.*, FIG. 4B, which illustrates circuitry for implementing a *umulxck* instruction; pp. 2-3, paragraph [1007]; p. 3 paragraph [1009]; and pp. 13-14, paragraph [1072] and Table 2.) The second arithmetic structures are further configured to generate a first partial result of a currently executing instance of the arithmetic instruction. (*See, e.g.*, FIG. 4B; and pp. 13-14, paragraph [1072] and Table 2.) The arithmetic instruction does not include an explicit source operand for specifying the high order bits. (*See, e.g.*, pp. 13-14, paragraph [1072] and Table 2; and pp. 14-15, paragraph [1076].) The first partial result represents the high order bits summed with low order bits of a multiplication result of the multiplication operation and with the third number. (*See, e.g.*, pp. 2-3, paragraph [1007].) The processor further includes a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application. (*See, e.g.*, FIG. 4B, extended carry register (exc) 403;pp. 13-14, paragraph [1072] and Table 2.)

Independent claim 66 is directed to an apparatus configured to support a public-key cryptography application. (*See, e.g.*, the Title; p. 1, paragraph [1002]; and pp. 7-8, paragraph [1056].) The apparatus includes means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, generated by a first arithmetic circuit, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set. (*See, e.g.*, FIG. 4A, which illustrates circuitry for implementing a *umulxc* instruction; p. 2, paragraph [1006]; p. 3 paragraph [1008]; and pp. 11-12, paragraph [1068].) The apparatus also includes means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction. (*See, e.g.*, FIG. 4A; pp. 11-12, paragraph [1068] and Table 1.) The currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits. (*See, e.g.*, p. 12, Table 1; and pp. 12-13, paragraph 1070].) The first partial result represents the high order bits of the previously executed single arithmetic instruction that are summed with low order bits of a multiplication result of a first number multiplied by a second number. (*See, e.g.*, p. 2, paragraph [1006].) The apparatus also includes means for using the first partial result in a subsequent computation in the public-key cryptography application. (*See, e.g.*, FIG. 4A, extended carry register (exc) 403; pp. 11-12, paragraphs [1068-1069] and Table 1.)

Independent claim 67 is directed to an apparatus configured to support a public-key cryptography application. (*See, e.g.*, the Title; p. 1, paragraph [1002]; and pp. 7-8, paragraph [1056].) The apparatus includes means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, from a first arithmetic circuit that generated the high order bits, to a second arithmetic circuit generating low order bits of a currently single executing arithmetic instruction of the processor instruction set. The apparatus also includes means for supplying a third number to the second arithmetic circuit. (*See, e.g.*, FIG. 4B, which illustrates circuitry for implementing a *umulxck* instruction; pp. 2-3, paragraph [1007]; p. 3 paragraph [1009]; and pp. 13-14, paragraph [1072] and Table 2.) The apparatus also includes means for using the second arithmetic circuit to generate a first partial result of the

currently executing single arithmetic instruction. (*See, e.g.*, FIG. 4B; and pp. 13-14, paragraph [1072] and Table 2.) The currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits. (*See, e.g.*, pp. 13-14, paragraph [1072] and Table 2; and pp. 14-15, paragraph [1076].) The first partial result represents the high order bits of the previously executed single arithmetic instruction summed with low order bits of a result of a first number multiplied by a second number and with the third number. (*See, e.g.*, pp. 2-3, paragraph [1007].) The apparatus also includes means for using the first partial result in a subsequent computation in the public-key cryptography application. (*See, e.g.*, FIG. 4B, extended carry register (exc) 403;pp. 13-14, paragraph [1072] and Table 2.)

The summary above describes various examples and embodiments of the claimed subject matter; however, the claims are not necessarily limited to any of these examples and embodiments. The claims should be interpreted based on the wording of the respective claims.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 4-10, 19, 21-26, 36, 38-42, 48, 52-60, 62, 66 and 67 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable Huppenthal, et al. (U.S. Patent 6,339,819) (hereinafter "Huppenthal") in view of Hinds, et al. (U.S. Patent 6,542,916) (hereinafter "Hinds"), and further in view of Chen et al. (U.S. Patent 6,763,365) (hereinafter "Chen").

2. Claims 2, 3, 15-18, 27-29, 35 and 43-46 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Lasher et al. (U.S. Patent 4,863,247) (hereinafter "Lasher")

3. Claims 11, 20, 30, 31, 37, 47 and 61 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Stribaek et al. (U.S. Patent 7,181,484) (hereinafter "Stribaek")

4. Claims 12-14, 32-34, 49-51 and 63-65 stand finally rejected under 35 U.S.C. § 103(a) as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Chen et al. (U.S. Patent 6,687,725) (hereinafter "Chen2")

VII. ARGUMENT

The Examiner rejected claims 1, 4-10, 19, 21-26, 36, 38-42, 48, 52-60, 62, 66 and 67 under 35 U.S.C. § 103(a) as being unpatentable over Huppenthal in view of Hinds, and further in view of Chen, claims 2, 3, 15-18, 27-29, 35 and 43-46 as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Lasher, claims 11, 20, 30, 31, 37, 47 and 61 as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Stribaek, and claims 12-14, 32-34, 49-51 and 63-65 as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Chen2. Appellants traverse the rejection of these claims for at least the following reasons. Different groups of claims are addressed under their respective subheadings.

First Ground of Rejection

The Examiner rejected claims 1, 4-10, 19, 21-26, 36, 38-42, 48, 52-60, 62, 66 and 67 under 35 U.S.C. § 103(a) as being unpatentable over Huppenthal in view of Hinds, and further in view of Chen.

Claims 1 and 4:

1. **The cited art clearly fails to teach or suggest** ***a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures*****; and*** ***the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a result of***

a first number multiplied by a second number, the summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit.

In rejecting Appellants' independent claims, the Examiner has repeatedly suggested that the claims are directed to the chaining of individual arithmetic operations into a single arithmetic instruction or the chaining of individual arithmetic operations into a single arithmetic instruction "to form a single arithmetic instruction." In these and other remarks, the Examiner has ignored the plain language of Appellants' claims, which requires that the processor include *in its instruction set* a single arithmetic instruction that meets the limitations recited in the claims. It appears that the Examiner does not appreciate the concept of a *processor instruction set*, which is an extremely well understood and basic feature of any processor, as defined by the processor's native instructions. It is also clear from the Title, Abstract, **plain language of the claim**, and Detailed Description that Appellants' claimed invention is directed to various embodiments of a specific processor instruction, not to a collection of processor instructions that perform the operations recited in Appellants' claims.

Appellants' claims also require that feedback be implicitly implemented between different instances of the specific processor instruction by the circuitry that implements the specific processor instruction. **Appellants assert that none of the processors described in any of the evidence of record include such an instruction or any implicit feedback between different instances of such an instruction.**

For example, on p. 2 of the Final Action of July 22, 2010, the Examiner submits, "Applicant argues that the referenced prior art does not disclose, add-chaining operation and multiply-chaining operation. The add-chaining and multiply-chaining operation discloses a combination (chaining) of accumulate and multiplication operations into a single arithmetic instruction. The Specification in paragraph [1076] states that the instruction performs multiply-accumulate-chaining operation, which combines (appears

to be equivalent to prior art chaining) add-chaining and multiply-chaining in one operation in order to avoid the multiplier latency from the processing of operand(s) information between arithmetic instructions."

However, paragraph [1076] of Appellants' Specification actually states, in its entirety, the following:

> The umulxck instruction is an efficient way to support public-key computations. Back-to-back scheduling of multi-word multiplications and accumulations is often difficult using today's instruction sets due to the multiplier latency. The umulxck instruction performs the multiply-accumulate-chaining operation which combines add-chaining and multiply-chaining in one operation and avoids the multiplier latency. Using the umulxck instruction, and referring again to FIG. 5, the calculation of a 64x1024 bit partial product (y0*X) and the accumulation with a previous partial product (s16:s0), can be accomplished in the following 20 instructions:

```
set register k=y0;
umulxck 0,0;                // clear extended-carry register exc first
r0 = umulxck x0, s0;
 . . .
 . . .
. . .
r15 = umulxck x15, s15;
r16 = umulxck 0, s16;       // catch 64 carryout bits
r17 = umulxck 0, 0;         // catch last carryout bit
```

As described in Appellants' specification, each single instance of the umulxck instruction in this example performs both multiply and accumulate operations and also implicitly allows for accumulating an additional partial result of a previous single umulxck instruction, i.e. without requiring additional add operations or specifying an additional operand for the additional accumulate operation. Specifically, a umulxck instruction of the form shown above (rd = umulxck rs1, rs2 - where rs1 and rs2 are source operands and rd is a destination operand), performs (rs1*k + rs2 + a partial result of a previous single arithmetic instruction, such as a previous umulxck instruction) without explicitly specifying the additional operand (the partial result of the previous single arithmetic instruction). The result register specified by operand rd receives the lower order bits of the result of this calculation, and the extended carry register (which is also

not specified by any of the operands of the instruction) receives the upper order bits of the result of this calculation.

In the example referenced by the Examiner, the third single arithmetic instruction (r0 = umulxck x0, s0):

performs the calculations (x0*k + s0 + exc), where exc = 0 (because the second instruction above cleared exc)
stores the lower order bits of the result of this calculation in r0; and
stores the upper order bits of the result of this calculation in exc.

In this example, the fourth single arithmetic instruction (not shown, but implied to be (r1 = umulxck x1, s1):

performs the calculations (x1*k + s1 + exc), where exc now contains the upper order bits of the result of the calculation performed for the third instruction, as noted above;
stores the lower order bits of the result of this calculation in r0; and
stores the upper order bits of the result of this calculation in exc.

Similarly, each successive instance of the single umulxck instruction in this example performs a similar calculation on the explicitly specified source operands and also implicitly adds the contents of exc that were stored in exc by the previous umulxck instruction as the high order bits of the calculation made for that previous umulxck instruction.

As described in detail above, the Examiner's characterization of the teachings of Appellants' Specification is clearly incorrect. The cited portions of Appellants' Specification describe a specific processor instruction (umulxck). When a single instance of this specific processor instruction is executed by the processor, the processor performs specific multiply and accumulate calculations, including the implicit addition of a partial result of a previously executed single instance of the umulxck instruction. In other words, as noted above, execution of the processor instructions recited in Appellants' claims facilitates a feedback relationship between two successive instances of a single arithmetic instruction, not merely a collection of arithmetic operations that are performed in response to a single initiated instruction, or a general concept of operation chaining in

performing a single arithmetic instruction. Because execution of these instructions facilitates this feedback relationship, the chaining of successive single instances of this umulxck instruction can be used to perform (and accelerate) the multi-word multiplies and accumulates that are common in cryptography applications.

In another example, on p. 3 of the Final Action of July 22, 2010, the Examiner also submits, "Huppenthal discloses an architecture for chaining a number of arithmetic operations (such as multiplication, accumulation, and etc) to form a single arithmetic instruction. The single arithmetic instruction is initiated and the sequence of chained operations is performed with operand transfer controlled by the computing system via the usage of a chain port mechanism. The chain port mechanism supplies operands to each successive arithmetic operation in the sequence. The chain mechanism supplies operands without any support from the processor. Hinds discloses the generation of a partial result (high-order or low-order bits). Huppenthal and Hinds disclose chaining a set of two or more arithmetic operations within a single arithmetic instruction and the generation of a partial result as the operand that is implicitly transferred between the arithmetic instructions." Appellants assert that the Examiner is clearly mischaracterizing the teachings of both Huppenthal and Hinds in suggesting that the combination teaches Appellants' claimed invention, according to the limitations recited in the claim itself.

Huppenthal is directed to a multiprocessor computer architecture incorporating a number of memory algorithmic processors ("MAP") in the memory subsystem or closely coupled to other processing elements (e.g., one or more fixed instruction set microprocessor-based processors) to enhance overall system processing speed. The memory algorithmic processor architecture is an assembly that contains field programmable gate arrays (FPGAs) functioning as the memory algorithmic processors. In other words, each memory algorithmic processor includes one or more FPGAs that are configurable to perform various functions that are not performed by one of the instructions of the fixed instruction microprocessor-based processors in the system. As described in the Summary section of Huppenthal, a MAP element can function autonomously from its host system (i.e. without processor intervention) once its operands

have been loaded. MAP elements (which are not processor instructions, nor do they represent circuitry to execute arithmetic instructions of a microprocessor instruction set) can be chained together to forward operands from one to another when multiple MAP elements are working together to perform a very large function. As described in detail in Huppenthal, the MAP architecture, and FPGAs thereof, perform functions in response to commands written to them using a write instruction that specifies a command and operands in the data. This is clearly not analogous to the limitations of Appellants' claims, which require circuits for executing multiple instances of a single arithmetic instruction of a processor instruction set. For example, Table 2 of Huppenthal (and the accompanying description) describe commands that are communicated to the MAP architecture by issuing a write instruction from processor 12. These commands are used to configure various elements of the MAP architecture (e.g., RMB, RUC, RECON, LDROM), to pass operands to the MAP architecture (e.g., WRTOP, LASTOP), and to initiate the performance of a function implemented in the MAP circuitry (e.g., START).

While performing a function using the MAP circuitry of Huppenthal may involve performing a series of calculations or other functions using one or more FPGAs, this teaches absolutely nothing about Appellants' claimed invention. Appellants' claims are not directed to the general concept of performing a collection of arithmetic operations to carry out a single arithmetic instruction, as the Examiner implies, or to the chaining of operations performed within a MAP architecture or FPGA in response to a command issued using one or more write instructions of a processor, but to specific arithmetic instructions in a processor's instruction set, individual instances of which implicitly add partial results of a previously executed single arithmetic instruction when executing a current single arithmetic instruction without explicitly specifying the partial result as an operand of the current single arithmetic instruction. The Examiner appears to be attempting to distill Appellants' invention down to a general "concept" or "gist" while ignoring the specific language of the claim. "Distilling an invention down to the 'gist' or 'thrust' of an invention disregards the requirement of analyzing the subject matter 'as a whole.'" *W.L. Gore & Assoc., Inc. v. Garlock, Inc.*, 721 F.2d 1540, 220 USPQ 303 (Fed. Cir. 1983), *cert. denied*, 469 U.S. 851 1984.

In yet another example, on pp. 3-4 of the Final Action of July 22, 2010, the Examiner submits, "Huppenthal and Hinds disclose the concept of chaining of arithmetic operations. This chaining concept is part of the architecture of the computing system disclosed within the Huppenthal prior art claimed invention. The chain operation is a legitimate operation within the instruction set of the Huppenthal prior art." Appellants again assert that the Examiner does not appear to understand the concept of a processor's instruction set. As discussed above, the instruction set of Huppenthal does not include a single arithmetic instruction that meets the limitations recited in Appellants' claim, nor does it include a "chain operation" as suggested by the Examiner. Instead, the operations referenced by the Examiner are invoked using a write instruction of the processor's instruction set.

On p. 5 of the Final Action of July 22, 2010, the Examiner submits, "Huppenthal discloses a method implemented in a device and storing the first partial result; and using the stored first partial result in a subsequent computation in the public-key cryptography application, the method comprising: a previously executed single arithmetic instruction of a processor set and wherein the currently executing single arithmetic instructions does not include an explicit source operand for specifying the high order bits". **Appellants assert that the Examiner has picked and chosen individual words and partial phrases taken from Appellants' claim in his remarks, completely ignoring the context in which they are used in the claim.** For example, the phrase "a method implemented in a device and storing the first partial result" has no meaning in the Examiner's remarks, as no partial result (or anything that could produce a result or partial result) is referenced by the Examiner. Similarly, the phrase "using the stored first partial result in a subsequent computation in the public-key cryptography application" has no meaning in the Examiner's remarks, since the Examiner has not identified a first computation with respect to the recited "subsequent computation." In addition, the phrase, "the method comprising: a previously executed single arithmetic instruction of a processor set" is incomprehensible, as it does not describe a method step at all. Finally, the phrase

"wherein the currently executing single arithmetic instructions does not include an explicit source operand for specifying the high order bits" has no meaning in the Examiner's remarks, since no such single arithmetic instructions or high order bits are referenced in his remarks.

The Examiner cites Huppenthal (in column 3, lines 1-7) as disclosing "number of MAP elements chained together to accomplish a single function or operation (implies a single arithmetic instruction)." **The Examiner's remarks regarding the implication of a single arithmetic instruction of a processor instruction set are completely unsupported by the reference itself.** In fact, as discussed above, the memory algorithmic processors taught by Huppenthal are explicitly disclosed as being separate from the fixed instruction set microprocessors in the system (such as processors 12_1 – 12_n). Instead, they act as co-processors implemented in FPGAs or another type of user array to perform algorithmic functions in response to commands issued to them using a write instruction of the fixed instruction set processor, which is clearly not an arithmetic instruction of the fixed instruction set processor.

The Examiner cites Huppenthal (in column 3, lines 18-25) as disclosing, "MAP elements can receive operands via chained port)" and (in column 18, line 66 – column 19, line 3) as disclosing, "output data from one MAP element to be sent directly to the user array of the next MAP element with no processor intervention via a chain port." Appellants again assert that the chaining of MAP elements to perform a large function coded in FPGAs and initiated by a write instruction teaches nothing about the limitations of Appellants' claim. For example, it does not teach circuitry that feeds back partial results of one arithmetic instruction of a processor's instruction set to circuitry executing another arithmetic instruction of the processor's instruction set. The chaining of MAP elements and passing of operands between MAP elements over a chain port when performing a single large operation implemented in one or more FPGAs, as described by Huppenthal, teach nothing about feedback between arithmetic instructions of a processor's instruction set.

The Examiner admits that Huppenthal does not specifically disclose a multiplying and summing sequence and relies on Hinds to teach such a sequence. Appellants again note that claim 1 does not merely require "a multiplying and summing sequence" but also requires feedback between two different instances of a single arithmetic instruction of a processor instruction set. The Examiner submits that Hinds discloses a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of an executed arithmetic instruction in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures and the second arithmetic circuit generating a first partial result of a currently executing arithmetic instruction in the public-key cryptography application, the first partial result representing the high order bits summed with low order bits of a result of a first number multiplied by a second number, the summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit. **Again, the Examiner has selectively ignored some of the words explicitly recited in the limitations of Appellants' claim, changing the context and meaning of these limitations.** For example, claim 1 does not merely recite "feeding back high order bits of an executed single arithmetic instruction" but requires circuitry for "feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set." Similarly, claim 1 does not merely recite "the second arithmetic circuit generating a first partial result of a currently executing arithmetic instruction in the public-key cryptography application" but requires the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set and that the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits.

The Examiner cites Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results:

high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." Hinds is directed to an apparatus and method for implementing a floating point MAC instruction that takes three operands as inputs. The cited portions of Hinds describe the operation of an individual floating point MAC instruction, which includes both a multiplication and an addition operation. As described in Hinds, all three operands of this MAC instruction are explicitly specified for the instruction; there is no additional operand that is implicitly added during execution of Hinds' MAC instruction. The Examiner's citation in column 8 refers to a prior art "chained" multiply-accumulate FPU, illustrated in FIG. 3B, which is operable to implement a single floating point multiply-accumulate operation (A+(B*C)) or floating-point multiply-subtract operation (-A+(B*C)) in response to issuance of a single floating-point instruction. This "chained" multiply-accumulate FPU passes results of a partial multiplier to a carry save adder and final product adder as part of executing a single floating point MAC instruction. This partial result is not implicitly fed back to a circuit implementing a subsequent arithmetic instruction, as in Appellants' claimed invention, nor is it fed back to a circuit implementing the current instruction from a circuit that generated a partial result of a previously executed instruction.

Since the floating point MAC instruction of Hinds does not implicitly add a partial result from the previous execution of another floating point MAC (or a partial result from the previous execution of any other arithmetic instruction), as suggested by the Examiner and as required by Appellant' claim, the combination of Huppenthal and Hinds clearly would not result in Appellants' claimed invention. At most, it could result in a system in which the floating point MAC instruction of Hinds (which does not include such feedback) is implemented algorithmically in one or more FPGAs of the MAP architecture of Huppenthal, e.g., if such an instruction is not included in the instruction set of the fixed-instruction processors of Huppenthal.

The Examiner admits that Huppenthal-Hinds does not specifically disclose supporting a cryptography application and relies on Chen to disclose supporting a public-key cryptography application (citing Chen, column 6, lines 23-25, and "arithmetic

operations to support acceleration of cryptographic functions"). Appellants assert, however, that Chen does not overcome the deficiencies of Huppenthal-Hinds in teaching Appellants' claimed invention. Accordingly, a *prima facie* rejection has not been established and the rejection includes clear error.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

Appellants note that the Examiner has not provided any reason to combine the Huppenthal and Hinds references. As stated in *KSR Int'l Co. v. Teleflex Inc.*, No. 04-1350, slip. op. at 14 (U.S. Apr. 30, 2007), "rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal standard of obviousness." The Examiner must show that "there was an apparent reason to combine the known elements in the fashion claimed." *Id.* The Examiner's analysis "should be made explicit." *Id.* Since the Examiner has not articulated any reason to combine the references, the rejection is improper.

On p. 6 of the Final Action of July 22, 2010, the Examiner states, "It would have been obvious to one of ordinary skill in the art to modify Huppenthal-Hinds to support a cryptographic application as taught by Chen. One of ordinary skill in the art would have been motivated to employ the teachings of Chen to greatly improve the performance of cryptographic circuits. (Chen col. 5, lines 40-42)." The Examiner seems to imply that the reason to modify Huppenthal-Hinds to support cryptographic applications is to improve the performance of something that apparently does not exist in Huppenthal-Hinds (cryptographic circuits) for something that the Examiner admits that Huppenthal-Hinds does not support (i.e. cryptographic applications). Therefore, the Examiner has not provided a valid reason to combine the references. In addition, as described in detail above, the cited art does not teach all of the limitations of Appellants' claims, whether taken alone or in combination.

On p. 4 of the Final Action of July 22, 2010, the Examiner submits that the prior art references and the claimed invention are in the same field(s), e.g., "computing system instruction set processing" and "high performance computations". However, this alone does not constitute a valid reason to combine the references. Appellants again assert that the Examiner has not explicitly articulated a reason to combine elements of the cited references in a way that teaches Appellants' invention, as he is required to do. Accordingly, a *prima facie* rejection has not been established and the rejection includes clear error.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 1.

Claims 38, 42 and 52:

1. **The cited art clearly fails to teach or suggest *a first plurality of arithmetic structures configured to generate high order bits for an arithmetic instruction of the processor's instruction set in a public-key cryptography application that includes a multiplication operation; and a second plurality of arithmetic structures configured to generate low order bits of the arithmetic instruction; wherein the second arithmetic structures are further configured to receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic instruction in the public-key cryptography application and to generate a first partial result of a currently executing instance of the arithmetic instruction, wherein the arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation.***

These limitations are similar to those of claim 1 discussed above, and the Examiner rejected claim 38 for reasons similar to those discussed above. Appellants traverse the rejection of this claim for at least the reasons presented above regarding limitations in this claim that are similar to those in claim 1.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 16 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Hinds to disclose arithmetic instructions is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided any reason to combine the Huppenthal and Hinds references in teaching claim 1.

On. p. 16 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Chen to disclose supporting a cryptographic application is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided a valid reason to combine these references. Therefore, the rejection is improper.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 38.

Independent claim 66:

1. The cited art clearly fails to teach or suggest *means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, generated by a first arithmetic circuit, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set; means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits of the previously executed single arithmetic instruction that are summed with low order bits of a multiplication result of a first number multiplied by a second number; and means for using the first partial result in a subsequent computation in the public-key cryptography application.*

These limitations are similar to those of claim 1 discussed above, and the Examiner rejected claim 66 for reasons similar to those discussed above. Appellants traverse the rejection of this claim for at least the reasons presented above regarding limitations in this claim that are similar to those in claim 1.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 25 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Hinds to disclose arithmetic instructions is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided any reason to combine the Huppenthal and Hinds references in teaching claim 1.

On. p. 25 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Chen to disclose supporting a cryptographic application is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided a valid reason to combine these references. Therefore, the rejection is improper.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 66.

Claims 21, 22, and 24:

1. The cited art clearly fails to teach or suggest *a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit to a second arithmetic circuit comprising a second plurality of arithmetic structures; the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography*

application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits summed with low order bits of a result of a first number multiplied by a second number... the summing being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit.

As shown above, Independent claim 21 includes limitations similar to those recited in claim 1 and discussed above, and was rejected for reasons similar to those discussed above regarding claim 1. Therefore, Appellants traverse the rejection of this claim for at least the reasons presented above regarding limitations in this claim that are similar to those in claim 1.

2. The cited art clearly fails to teach or suggest *supplying a third number to the second arithmetic circuit* and *the first partial result being a representation of the high order bits summed with low order bits of a result of a first number multiplied by a second number and with the third number.*

The Examiner submits that Hinds discloses these limitations using the same citations and reasoning as those included in remarks directed to claim 1. However, as discussed in detail above, Hinds teaches a floating point MAC instruction of the form (A+(B*C)), for which operands A, B, and C are explicitly specified. Hinds does not disclose a MAC instruction that feeds back partial results of a previously executed instruction, the first partial result being a representation of the high order bits summed with low order bits of a result of a first number multiplied by a second number and with the third number, as required by claim 21.

3. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 11 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Hinds to disclose arithmetic instructions is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided any reason to combine the Huppenthal and Hinds references in teaching claim 1.

On. p. 12 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Chen to disclose supporting a cryptographic application is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided a valid reason to combine these references. Therefore, the rejection is improper.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 21.

Claims 53 and 57-60:

1. **The cited art clearly fails to teach or suggest** ***a first plurality of arithmetic structures configured to generate high order bits for an arithmetic instruction of the processor's instruction set in a public-key cryptography application that includes a multiplication operation of a first and a second number; and a second plurality of arithmetic structures configured to generate low order bits of the arithmetic instruction; wherein the second arithmetic structures are configured to: receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic instruction; receive a third number; and generate a first partial result of a currently executing instance of the arithmetic instruction, wherein the arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation and with the third number.***

As shown above, Claim 53 includes limitations similar to those recited in claims 1 and 21 and discussed above, and was rejected for the same reasons as claims 1 and 21.

Therefore, Appellants traverse the rejection of this claim for at least the reasons presented above regarding limitations in this claim that are similar to those in claim 1 and 21.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 20 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Hinds to disclose arithmetic instructions is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided any reason to combine the Huppenthal and Hinds references in teaching claim 1.

On. p. 20 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Chen to disclose supporting a cryptographic application is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided a valid reason to combine these references. Therefore, the rejection is improper.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 53.

Independent claim 67:

1. The cited art clearly fails to teach or suggest *means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, from a first arithmetic circuit that generated the high order bits, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set; means for supplying a third number to the second arithmetic circuit; means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits of the previously executed single arithmetic*

instruction summed with low order bits of a result of a first number multiplied by a second number and with the third number; and means for using the first partial result in a subsequent computation in the public-key cryptography application.

As shown above, Claim 67 includes limitations similar to those recited in claims 1 and 21 and discussed above, and was rejected for the same reasons as claims 1 and 21. Therefore, Appellants traverse the rejection of this claim for at least the reasons presented above regarding limitations in this claim that are similar to those in claim 1 and 21.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

The Examiner has not provided any remarks at all regarding a reason to combine the Huppenthal and Hinds references in teaching claim 67.

On. p. 26 of the Final Action of July 22, 2010, the Examiner states, "Motivation for Chen to disclose supporting a cryptographic application is as stated in Claim 1 above." However, as discussed above, the Examiner has not provided a valid reason to combine these references. Therefore, the rejection is improper.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 67.

Dependent claim 5:

1. The cited art clearly fails to teach or suggest *generating a second partial result of the currently executing single arithmetic instruction in the first arithmetic circuit, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number.*

In rejecting claim 5, the Examiner submits that Hinds discloses these limitations, citing Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results: high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." As discussed above, the cited portions of Hinds describe the operation of an individual floating point MAC instruction, and a prior art "chained" multiply-accumulate FPU that implements a single floating point MAC operation, and does not describe generating (or feeding back) the first partial result of Appellants' claims. Appellants further assert that Hinds does not disclose the specific limitations of this claim regarding the generation of a second partial result.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 7 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 5.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 5.

Dependent claim 23:

Dependent claim 23 recites limitations similar to those of claim 5 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 39:

Dependent claim 39 recites limitations similar to those of claim 5 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 54:

Dependent claim 54 recites limitations similar to those of claim 5 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 6:

1. **The cited art clearly fails to teach or suggest *generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed single arithmetic instruction.***

In rejecting claim 6, the Examiner submits that Hinds discloses these limitations, again citing Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results: high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." As discussed above, the cited portions of Hinds describe the operation of an individual floating point MAC instruction, and a prior art "chained" multiply-accumulate FPU that implements a single floating point MAC operation, and does not describe generating (or feeding back) the first partial result of Appellants'

claims. Appellants further assert that Hinds does not disclose the specific limitations of this claim regarding the generation of a second partial result.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 8 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 6.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 6.

Dependent claim 7:

1. The cited art clearly fails to teach or suggest *supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.*

In rejecting claim 7, the Examiner submits that Hinds discloses these limitations, again citing Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results: high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." As discussed above, the cited portions of Hinds describe the operation of an individual floating point MAC instruction, and a prior art "chained" multiply-accumulate FPU that implements a single floating point MAC operation, and

does not describe generating (or feeding back) the first partial result of Appellants' claims, or generating a second partial result. Appellants further assert that Hinds does not disclose the specific limitations of this claim regarding supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating those first and second partial results.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 8 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 7.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 7.

Dependent claim 25:

Dependent claim 25 recites limitations similar to those of claim 7 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 40:

Dependent claim 40 recites limitations similar to those of claim 7 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 55:

Dependent claim 55 recites limitations similar to those of claim 7 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 8:

1. The cited art clearly fails to teach or suggest *wherein the generating of the first and second partial result is in response to execution of the currently executing single arithmetic instruction.*

In rejecting claim 8, the Examiner submits that Hinds discloses these limitations, again citing Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results: high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." As discussed above, the cited portions of Hinds describe the operation of an individual floating point MAC instruction, and a prior art "chained" multiply-accumulate FPU that implements a single floating point MAC operation, and does not describe generating (or feeding back) the first partial result of Appellants' claims, or generating a second partial result. Appellants further assert that Hinds does not disclose the specific limitations of this claim regarding generating those first and second partial results in response to execution of the single arithmetic instruction of Appellants' claims. No such single arithmetic instruction (one meeting the limitations of Appellants' claims) is taught by Huppenthal-Hinds-Chen.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 9 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 8.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 8.

Dependent claim 26:

Dependent claim 26 recites limitations similar to those of claim 8 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 41:

Dependent claim 41 recites limitations similar to those of claim 8 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 56:

Dependent claim 56 recites limitations similar to those of claim 8 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 9:

1. The cited art clearly fails to teach or suggest *wherein the generating of the first and second partial result is in response to execution of the currently executing single arithmetic instruction*.

In remarks directed to claim 9, the Examiner states that Hinds discloses the limitations of claim 9, without explaining how or why he believes this is the case. However, as discussed above in remarks directed to claim 8, Hinds does not disclose the specific limitations of this claim regarding generating the first and second partial results of Appellants' claims in response to execution of the single arithmetic instruction of Appellants' claims. No such single arithmetic instruction (one meeting the limitations of Appellants' claims) is taught by Huppenthal-Hinds-Chen.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 9 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 9.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 9.

Dependent claim 10:

1. The cited art clearly fails to teach or suggest *wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.*

In rejecting claim 10, the Examiner submits that Hinds discloses these limitations, again citing Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results: high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." As discussed above, the cited portions of Hinds describe the operation of an individual floating point MAC instruction, and a prior art "chained" multiply-accumulate FPU that implements a single floating point MAC operation, and does not describe generating (or feeding back) the first partial result of Appellants' claims, or generating a second partial result. Appellants further assert that Hinds does not disclose the specific limitations of this claim regarding a plurality of carry save adder tree columns, much less that such carry save adder tree columns are included in first and second arithmetic structures meeting the limitations of Appellants' claims.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 10 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 10.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 10.

Dependent claim 48:

Dependent claim 48 recites limitations similar to those of claim 10 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 62:

Dependent claim 62 recites limitations similar to those of claim 10 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Dependent claim 19:

1. The cited art clearly fails to teach or suggest *feeding back high order bits of the currently executing single arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction of the processor instruction set.*

In rejecting claim 19, the Examiner submits that Hinds discloses these limitations, again citing Hinds (in column 3, lines 5-17) as disclosing, "applying a multiply-accumulate operation to operands; multiplying operands and adding multiplication results" and (in column 8, lines 6-25) as disclosing, "chained multiply-accumulate operation; carry save adders and usage of partial multiplier (partial results: high order bits); output of results of partial multiplier is normalized and passed to carry save adders and final product adder)." As discussed above, the cited portions of Hinds describe the operation of an individual floating point MAC instruction, and a prior art "chained" multiply-accumulate FPU that implements a single floating point MAC operation, and does not describe feeding back high order bits of a currently executing single arithmetic instruction for use with execution of a subsequent single arithmetic instruction. Instead,

the feedback described by Hinds is internal to the execution of a single floating point MAC for which all three of its operands are explicitly specified.

2. The Examiner has not stated a proper reason to combine the teachings of the cited art.

On. p. 10 of the Final Action of July 22, 2010, the Examiner merely states, "Motivation for Hinds to disclose arithmetic operations is as stated in Claim 1 above." This statement clearly does not articulate any reason for combining Huppenthal and Hinds (much less Huppenthal, Hinds, and Chen) in teaching the specific limitations of Appellants' claim 19.

For at least the reasons stated above, Appellants assert that the Examiner has failed to establish a *prima facie* rejection of claim 19.

Dependent claim 36:

Dependent claim 36 recites limitations similar to those of claim 19 and was rejected for similar reasons. Therefore, the arguments presented above apply with equal force to this claim, as well.

Second Ground of Rejection

The Examiner rejected claims 2, 3, 15-18, 27-29, 35 and 43-46 as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Lasher. Appellants traverse the rejection of these claims for at least the reasons presented above regarding the independent claims from which they depend.

Third Ground of Rejection

The Examiner rejected claims 11, 20, 30, 31, 37, 47 and 61 as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Striback. Appellants traverse the rejection of these claims for at least the reasons presented above regarding the independent claims from which they depend.

Fourth Ground of Rejection

The Examiner rejected claims 12-14, 32-34, 49-51 and 63-65 as being unpatentable over Huppenthal, Hinds and Chen, and further in view of Chen2. Appellants traverse the rejection of these claims for at least the reasons presented above regarding the independent claims from which they depend.

CONCLUSION

For the foregoing reasons, it is submitted that the Examiner's rejection of claims 1-67 was erroneous, and reversal of his decision is respectfully requested.

The Commissioner is authorized to charge the appeal brief fee and any other fees that may be due to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/6000-31500/RCK.

Respectfully submitted,

/ Robert C. Kowert /
Robert C. Kowert, Reg. #39,255
Agent for Appellants

Meyertons, Hood, Kivlin, Kowert & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
(512) 853-8850

Date: January 24, 2011

VIII. CLAIMS APPENDIX

The claims on appeal are as follows.

1. A method implemented in a device supporting a public-key cryptography application, the method comprising:

a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit, to a second arithmetic circuit comprising a second plurality of arithmetic structures;

the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a result of a first number multiplied by a second number, the summing of the high order bits being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit;

storing the first partial result; and

using the stored first partial result in a subsequent computation in the public-key cryptography application.

2. The method as recited in claim 1 wherein the high order bits are fed back in redundant number representation.

3. The method as recited in claim 2 wherein the redundant number representation includes sum and carry bits.

4. The method as recited in claim 1, further comprising feeding back the high order bits through a register to the second arithmetic circuit.

5. The method as recited in claim 1, further comprising:

generating a second partial result of the currently executing single arithmetic instruction in the first arithmetic circuit, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number.

6. The method as recited in claim 1, further comprising:

generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed single arithmetic instruction.

7. The method as recited in claim 6 further comprising supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

8. The method as recited in claim 5 wherein the generating of the first and second partial result is in response to execution of the currently executing single arithmetic instruction.

9. The method as recited in claim 6 wherein the generating of the first and second partial result is in response to execution of the currently executing single arithmetic instruction.

10. The method as recited in claim 1, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.

11. The method as recited in claim 1, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns.

12. The method as recited in claim 1, wherein at least one of the first and second pluralities of arithmetic structures is usable to perform both integer and XOR multiplication.

13. The method as recited in claim 12, further comprising a logical circuit in at least one of the first and second arithmetic circuits supplying a fixed value if in XOR multiplication mode or a variable value that varies according to inputs supplied to the logical circuit if in integer multiplication mode, to thereby ensure a result is determined in XOR multiplication unaffected by carry logic performing carries in integer multiplication mode.

14. The method as recited in claim 13 wherein the logical circuit operates as a majority circuit in integer multiplication mode and outputs a zero in the XOR multiplication mode.

15. The method as recited in claim 1 wherein the first partial result is in redundant number representation.

16. The method as recited in claim 15 further comprising supplying the first partial result to an adder circuit to generate a non redundant representation of the first partial result and a carry out value.

17. The method as recited in claim 16 further comprising feeding back the carry out value to the adder circuit.

18. The method as recited in claim 16, further comprising feeding back the carry out value to the second arithmetic circuit.

19. The method as recited in claim 1, further comprising feeding back high order bits of the currently executing single arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction of the processor instruction set.

20. The method as recited claim 1 further comprising storing the high order bits into an extended carry register.

21. A method implemented in a device supporting a public-key cryptography application, the method comprising:

a first arithmetic circuit comprising a first plurality of arithmetic structures feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set in the public-key cryptography application, generated by the first arithmetic circuit to a second arithmetic circuit comprising a second plurality of arithmetic structures;

supplying a third number to the second arithmetic circuit;

the second arithmetic circuit generating a first partial result of a currently executing single arithmetic instruction of the processor instruction set in

the public-key cryptography application, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits summed with low order bits of a result of a first number multiplied by a second number and with the third number, the summing being performed during multiplication of the first number and the second number, the summing and at least a portion of the multiplication being performed in the second arithmetic circuit;

storing the first partial result; and

using the first partial result in a subsequent computation in the public-key cryptography application.

22. The method as recited in claim 21, further comprising feeding back the high order bits through a register to the second arithmetic circuit.

23. The method as recited in claim 21, further comprising:

the first arithmetic circuit generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number.

24. The method as recited in claim 21, further comprising:

generating a second partial result of the currently executing single arithmetic instruction, the second partial result representing the high order bits of the multiplication result of the first number multiplied by the second number summed with the high order bits of the previously executed single arithmetic instruction and the third number.

25. The method as recited in claim 24 further comprising supplying values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

26. The method as recited in claim 23 wherein the generating of the first and second partial result is in response to execution of the single arithmetic instruction.

27. The method as recited in claim 21

supplying the first partial result to an adder circuit to generate a non redundant representation of the first partial result and a carry out value.

28. The method as recited in claim 27 further comprising feeding back the carry out value to the adder circuit.

29. The method as recited in claim 27, method further comprising feeding back the carry out value to the second arithmetic circuit.

30. The method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns.

31. The method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.

32. The method as recited in claim 21, wherein at least one of the first and second pluralities of arithmetic structures is usable to perform both integer and XOR multiplication.

33. The method as recited in claim 32, further comprising a logic circuit in at least one of the first and second pluralities of arithmetic structures supplying a fixed value if in XOR multiplication mode or a variable value that varies according to inputs supplied to the logical circuit if in integer multiplication mode, to thereby ensure a result is determined in XOR multiplication unaffected by carry logic performing carries in integer multiplication mode.

34. The method as recited in claim 33 wherein the logic circuit operates as a majority circuit in integer multiplication mode and outputs a zero in the XOR multiplication mode.

35. The method as recited in claim 21 wherein the high order bits are in redundant number representation.

36. The method as recited in claim 21 further comprising feeding back high order bits of the currently executing single arithmetic instruction from the first arithmetic circuit to the second arithmetic circuit for use with execution of a subsequent single arithmetic instruction of the processor instruction set.

37. The method as recited in claim 21 further comprising storing the high order bits into an extended carry register.

38. A processor configured to support public-key cryptography applications, comprising:

a first plurality of arithmetic structures configured to generate high order bits for an arithmetic instruction of the processor's instruction set in a public-key cryptography application that includes a multiplication operation; and

a second plurality of arithmetic structures configured to generate low order bits of the arithmetic instruction;

wherein the second arithmetic structures are further configured to receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic instruction in the public-key cryptography application and to generate a first partial result of a currently executing instance of the arithmetic instruction, wherein the arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation; and

wherein the processor further comprises a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application.

39. The processor as recited in claim 38, wherein the first arithmetic structures are configured to generate a second partial result of the currently executing instance of the arithmetic instruction, the second partial result representing the high order bits of the currently executing instance of the arithmetic instruction.

40. The processor as recited in claim 39, wherein the second arithmetic structures are further configured to supply values generated in one or more most significant columns of the second arithmetic structures to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

41. The processor as recited in claim 39, wherein the first and second arithmetic structures are configured to generate the first and second partial results in response to execution of an instance of the arithmetic instruction.

42. The processor as recited in claim 38, further comprising a register coupled to the first and second arithmetic structures to supply the high order bits to the second arithmetic structures.

43. The processor as recited in claim 38, wherein the first partial result is in redundant number representation.

44. The processor as recited in claim 43, further comprising an adder circuit configured to receive the first partial result and to generate a non redundant representation of the first partial result and a carry out value.

45. The processor as recited in claim 44, wherein adder circuit is configured to feed the carry out value back to itself as an input.

46. The processor as recited in claim 44, wherein adder circuit is configured to feed the carry out value back to the second arithmetic structures.

47. The processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of Wallace tree columns.

48. The processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures comprises a plurality of carry save adder tree columns.

49. The processor as recited in claim 38, wherein at least one of the first and second pluralities of arithmetic structures is configured to selectively perform one of integer and XOR multiplication according to a control signal.

50. The processor as recited in claim 49, further comprising a plurality of logic circuits in the first and second pluralities of arithmetic structures, each logic circuit responsive to the control signal to supply a fixed output value in XOR multiplication

mode and a variable output value in integer multiplication mode, the variable output value varying according to values of inputs supplied to the logic circuit, to thereby ensure a result is determined in XOR multiplication mode unaffected by carry logic generating carries in integer multiplication mode.

51. The processor as recited in claim 50, wherein the logical circuit is configured to operate as a majority circuit in integer multiplication mode and to output a zero in XOR multiplication mode.

52. The processor as recited in claim 38, wherein the processor is a general purpose processor.

53. A processor configured to support public-key cryptography applications, comprising:

a first plurality of arithmetic structures configured to generate high order bits for an arithmetic instruction of the processor's instruction set in a public-key cryptography application that includes a multiplication operation of a first and a second number; and

a second plurality of arithmetic structures configured to generate low order bits of the arithmetic instruction;

wherein the second arithmetic structures are configured to:

receive the high order bits generated by the first plurality of arithmetic structures during execution of a previous instance of the arithmetic instruction;

receive a third number; and

generate a first partial result of a currently executing instance of the arithmetic instruction, wherein the arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits summed with low order bits of a multiplication result of the multiplication operation and with the third number; and

wherein the processor further comprises a register configured to store the first partial result for use in a subsequent arithmetic operation in the public-key cryptography application.

54. The processor as recited in claim 53, wherein the first arithmetic structures are further configured to generate a second partial result of the currently executing instance of the arithmetic instruction, the second partial result representing the high order bits of the currently executing instance of the arithmetic instruction.

55. The processor as recited in claim 54, wherein the second arithmetic structures are further configured to generate values in one or more most significant columns and to supply them to one or more least significant columns of the first arithmetic structures while generating the first and second partial results.

56. The processor as recited in claim 54, wherein the first arithmetic structures are configured to generate the first and second partial result in response to execution of an instance of the arithmetic instruction.

57. The processor as recited in claim 53, further comprising a register coupled to the first and second arithmetic structures to supply the high order bits to the second arithmetic structures.

58. The processor as recited in claim 53, further comprising an adder circuit configured to receive the first partial result and to generate a non redundant representation of the first partial result and a carry out value.

59. The processor as recited in claim 58 wherein the adder circuit is further configured to feed the carry out value back to itself as an input.

60. The processor as recited in claim 58, wherein the adder circuit is further configured to feed the carry out value back to the second arithmetic structures.

61. The processor as recited in claim 53, wherein at least one of the first and second arithmetic structures comprises Wallace tree columns.

62. The processor as recited in claim 53, wherein at least one of the first and second arithmetic structures comprises carry save adder tree columns.

63. The processor as recited in claim 53, wherein the arithmetic structures are configured to selectively perform one of integer and XOR multiplication according to a control signal.

64. The processor as recited in claim 63, further comprising a plurality of logic circuits in at least one of the first and second pluralities of arithmetic structures, each logic circuit responsive to the control signal to supply a fixed output value in XOR multiplication mode and a variable output value in integer multiplication mode, the variable output value varying according to values of inputs supplied to the logic circuit, to thereby ensure a result is determined in XOR multiplication mode unaffected by carry logic generating carries in integer multiplication mode.

65. The processor as recited in claim 64, wherein the logical circuit is configured to operate as a majority circuit in integer multiplication mode and to output a zero in the XOR multiplication mode.

66. An apparatus configured to support a public-key cryptography application, comprising:

means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, generated by a first arithmetic circuit, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set;

means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result representing the high order bits of the previously executed single arithmetic instruction that are summed with low order bits of a multiplication result of a first number multiplied by a second number; and

means for using the first partial result in a subsequent computation in the public-key cryptography application.

67. An apparatus configured to support a public-key cryptography application comprising:

means for feeding back high order bits of a previously executed single arithmetic instruction of a processor instruction set, from a first arithmetic circuit that generated the high order bits, to a second arithmetic circuit generating low order bits of a currently executing single arithmetic instruction of the processor instruction set;

means for supplying a third number to the second arithmetic circuit;

means for using the second arithmetic circuit to generate a first partial result of the currently executing single arithmetic instruction, wherein the currently executing single arithmetic instruction does not include an explicit source operand for specifying the high order bits, the first partial result being a representation of the high order bits of the previously executed single arithmetic instruction summed with low order bits of a result of a first number multiplied by a second number and with the third number; and

means for using the first partial result in a subsequent computation in the public-key cryptography application.

IX. EVIDENCE APPENDIX

No evidence submitted under 37 CFR §§ 1.130, 1.131 or 1.132 or otherwise entered by the Examiner is relied upon in this appeal.

X. RELATED PROCEEDINGS APPENDIX

There are no related proceedings.